SLEPc

A Study on SLEPc, a Library for Scalable Eigensolvers, and its Scientific and Engineering Applications

Jose E. Roman

High Performance Networking and Computing Group (GRyCAP)
Universidad Politécnica de Valencia, Spain

(joint work with V. Hernandez, A. Tomas, V. Vidal)

February, 2005

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

grycap
Grupo de Redes y Computación de Altas Prestaciones

Talk Outline

Overview of SLEPc

Eigenvalue Problems

Consider the following eigenvalue problems

Standard Eigenproblem

$$Ax = \lambda x$$

Generalized Eigenproblem

$$Ax = \lambda Bx$$

where

- λ is a (complex) scalar: *eigenvalue*
- x is a (complex) vector: *eigenvector*
- Matrices A and B can be real or complex
- Matrices A and B can be symmetric (Hermitian) or not
- Typically, B is symmetric positive (semi-) definite

Solution of the Eigenvalue Problem

There are n eigenvalues (counted with their multiplicities)

Partial eigensolution: nev solutions

$$\lambda_0, \lambda_1, \ldots, \lambda_{nev-1} \in \mathbb{C}$$
$$x_0, x_1, \ldots, x_{nev-1} \in \mathbb{C}^n$$

nev = number of eigenvalues / eigenvectors (eigenpairs)

Different requirements:

- Compute a few of the dominant eigenvalues (largest magnitude)
- Compute a few λ_i's with smallest or largest real parts
- Compute all λ_i's in a certain region of the complex plane

Solution of the Eigenvalue Problem

There are n eigenvalues (counted with their multiplicities)

Partial eigensolution: nev solutions

$$\lambda_0, \lambda_1, \ldots, \lambda_{nev-1} \in \mathbb{C}$$
$$x_0, x_1, \ldots, x_{nev-1} \in \mathbb{C}^n$$

nev = number of eigenvalues / eigenvectors (eigenpairs)

Different requirements:

- Compute a few of the dominant eigenvalues (largest magnitude)
- Compute a few λ_i's with smallest or largest real parts
- Compute all λ_i's in a certain region of the complex plane

Spectral Transformation

A general technique that can be used in many methods

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

In the transformed problem

- The eigenvectors are not altered
- The eigenvalues are modified by a simple relation
- Convergence is usually improved (better separation)

Shift of Origin

$$T_S = A + \sigma I$$

Shift-and-invert

$$T_{SI} = (A - \sigma I)^{-1}$$

Cayley

$$T_C = (A - \sigma I)^{-1}(A + \tau I)$$

Drawback: T not computed explicitly, linear solves instead

Spectral Transformation

A general technique that can be used in many methods

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

In the transformed problem

- The eigenvectors are not altered
- The eigenvalues are modified by a simple relation
- Convergence is usually improved (better separation)

Shift of Origin

$T_S = A + \sigma I$

Shift-and-invert

$T_{SI} = (A - \sigma I)^{-1}$

Cayley

$T_C = (A - \sigma I)^{-1}(A + \tau I)$

Drawback: T not computed explicitly, linear solves instead

Spectral Transformation

A general technique that can be used in many methods

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

In the transformed problem

- The eigenvectors are not altered
- The eigenvalues are modified by a simple relation
- Convergence is usually improved (better separation)

Shift of Origin

$$T_S = A + \sigma I$$

Shift-and-invert

$$T_{SI} = (A - \sigma I)^{-1}$$

Cayley

$$T_C = (A - \sigma I)^{-1}(A + \tau I)$$

Drawback: T not computed explicitly, linear solves instead

Observations to Be Considered

- Various problem characteristics: Problems can be real/complex, Hermitian/non-Hermitian
- Many formulations: not all eigenproblems are formulated as simply $Ax = \lambda x$ or $Ax = \lambda Bx$
- Many ways of specifying which solutions must be sought

Goal: provide a uniform, coherent way of addressing these problems

- Internally, solvers can be quite complex (deflation, restart, ...)
- Spectral transformations can be used irrespective of the solver
- Repeated linear solves may be required

Goal: hide eigensolver complexity and separate spectral transform

Observations to Be Considered

- Various problem characteristics: Problems can be real/complex, Hermitian/non-Hermitian
- Many formulations: not all eigenproblems are formulated as simply $Ax = \lambda x$ or $Ax = \lambda Bx$
- Many ways of specifying which solutions must be sought

Goal: provide a uniform, coherent way of addressing these problems

- Internally, solvers can be quite complex (deflation, restart, ...)
- Spectral transformations can be used irrespective of the solver
- Repeated linear solves may be required

Goal: hide eigensolver complexity and separate spectral transform

Observations to Be Considered

- Various problem characteristics: Problems can be real/complex, Hermitian/non-Hermitian
- Many formulations: not all eigenproblems are formulated as simply $Ax = \lambda x$ or $Ax = \lambda Bx$
- Many ways of specifying which solutions must be sought

Goal: provide a uniform, coherent way of addressing these problems

- Internally, solvers can be quite complex (deflation, restart, ...)
- Spectral transformations can be used irrespective of the solver
- Repeated linear solves may be required

Goal: hide eigensolver complexity and separate spectral transform

Observations to Be Considered

- Various problem characteristics: Problems can be real/complex, Hermitian/non-Hermitian
- Many formulations: not all eigenproblems are formulated as simply $Ax = \lambda x$ or $Ax = \lambda Bx$
- Many ways of specifying which solutions must be sought

Goal: provide a uniform, coherent way of addressing these problems

- Internally, solvers can be quite complex (deflation, restart, ...)
- Spectral transformations can be used irrespective of the solver
- Repeated linear solves may be required

Goal: hide eigensolver complexity and separate spectral transform

Executive Summary

SLEPc: Scalable Library for Eigenvalue Problem Computations

A *general* library for solving large-scale sparse eigenproblems on parallel computers

- For standard and generalized eigenproblems
- For real and complex arithmetic
- For Hermitian or non-Hermitian problems

Current version: 2.2.1 (released August 2004)

`http://www.grycap.upv.es/slepc`

SLEPc and PETSc

SLEPc extends PETSc for solving eigenvalue problems

PETSc: Portable, Extensible Toolkit for Scientific Computation

- Software for the solution of PDE's in parallel computers
- A freely available and supported research code
- Usable from C, C++, Fortran77/90
- Focus on abstraction, portability, interoperability, ...
- Object-oriented design (encapsulation, inheritance and polymorphism)
- Current: 2.2.1 `http://www.mcs.anl.gov/petsc`

SLEPc inherits all good properties of PETSc

Structure of SLEPc

SLEPc adds two new objects: EPS and ST

EPS: Eigenvalue Problem Solver

- The user specifies the problem via this object (entry point to SLEPc)
- Provides a collection of eigensolvers
- Allows the user to specify a number of parameters (e.g. which portion of the spectrum)

ST: Spectral Transformation

- Used to transform the original problem into $Tx = \theta x$
- Always associated to an EPS object, not used directly

Structure of SLEPc

SLEPc adds two new objects: EPS and ST

EPS: Eigenvalue Problem Solver

- The user specifies the problem via this object (entry point to SLEPc)
- Provides a collection of eigensolvers
- Allows the user to specify a number of parameters (e.g. which portion of the spectrum)

ST: Spectral Transformation

- Used to transform the original problem into $Tx = \theta x$
- Always associated to an EPS object, not used directly

Structure of SLEPc

SLEPc adds two new objects: EPS and ST

EPS: Eigenvalue Problem Solver

- The user specifies the problem via this object (entry point to SLEPc)
- Provides a collection of eigensolvers
- Allows the user to specify a number of parameters (e.g. which portion of the spectrum)

ST: Spectral Transformation

- Used to transform the original problem into $Tx = \theta x$
- Always associated to an EPS object, not used directly

SLEPc/PETSc Diagram

PETSc

Nonlinear Solvers

Newton–based Methods		Other
Line Search	Trust Region	

Time Steppers

Euler	Backward Euler	Pseudo Time Stepping	Other

Krylov Subspace Methods

GMRES	CG	CGS	Bi–CGStab	TFQMR	Richardson	Chebychev	Other

Preconditioners

Additive Schwarz	Block Jacobi	Jacobi	ILU	ICC	LU	Other

Matrices

Compressed Sparse Row (AIJ)	Blocked Compressed Sparse Row (BAIJ)	Block Diagonal (BDIAG)	Dense	Other

Vectors

Index Sets

Indices	Block Indices	Stride	Other

SLEPc

Eigensolvers

Power/RQI	Subspace	Arnoldi
Arpack	Blzpack	Other

Spectral Transform

Shift	Shift–and–invert	Cayley

SLEPc Highlights

- Growing number of eigensolvers
- Seamlessly integrated spectral transformation
- Easy programming with PETSc's object-oriented style
- Data-structure neutral implementation
- Run-time flexibility, giving full control over the solution process
- Portability to a wide range of parallel platforms
- Usable from code written in C, C++ and Fortran
- Extensive documentation

Basic Usage

Basic Usage

Usual steps for solving an eigenvalue problem with SLEPc:

1. Create an EPS object
2. Define the eigenvalue problem
3. (Optionally) Specify options for the solution
4. Run the eigensolver
5. Retrieve the computed solution
6. Destroy the EPS object

All these operations are done via a generic interface, common to all the eigensolvers

Simple Example

```
EPS          eps;       /* eigensolver context   */
Mat          A, B;      /* matrices of Ax=kBx    */
Vec          xr, xi;    /* eigenvector, x        */
PetscScalar  kr, ki;    /* eigenvalue, k         */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);

EPSSolve(eps);

EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}

EPSDestroy(eps);
```

Simple Example

```
EPS        eps;       /*  eigensolver context  */
Mat        A, B;      /*  matrices of Ax=kBx   */
Vec        xr, xi;    /*  eigenvector, x       */
PetscScalar kr, ki;   /*  eigenvalue, k        */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);

EPSSolve(eps);

EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}

EPSDestroy(eps);
```

Simple Example

```
EPS          eps;      /*  eigensolver context   */
Mat          A, B;     /*  matrices of Ax=kBx    */
Vec          xr, xi;   /*  eigenvector, x        */
PetscScalar kr, ki;    /*  eigenvalue, k         */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);

EPSSolve(eps);

EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}

EPSDestroy(eps);
```

SLEPc

Simple Example

```
EPS          eps;       /* eigensolver context  */
Mat          A, B;      /* matrices of Ax=kBx   */
Vec          xr, xi;    /* eigenvector, x       */
PetscScalar kr, ki;     /* eigenvalue, k        */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);

EPSSolve(eps);

EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}

EPSDestroy(eps);
```

Simple Example

```
EPS         eps;       /* eigensolver context  */
Mat         A, B;      /* matrices of Ax=kBx   */
Vec         xr, xi;    /* eigenvector, x       */
PetscScalar kr, ki;    /* eigenvalue, k        */

EPSCreate(PETSC_COMM_WORLD, &eps);
EPSSetOperators(eps, A, B);
EPSSetProblemType(eps, EPS_GNHEP);
EPSSetFromOptions(eps);

EPSSolve(eps);

EPSGetConverged(eps, &nconv);
for (i=0; i<nconv; i++) {
  EPSGetEigenpair(eps, i, &kr, &ki, xr, xi);
}

EPSDestroy(eps);
```

Details: Solving the Problem

`EPSSolve(EPS eps)`

Launches the eigensolver

Currently available eigensolvers:

- Power Iteration with deflation. This includes:
  - Inverse Iteration
  - Rayleigh Quotient Iteration (RQI)
- Subspace Iteration with Rayleigh-Ritz projection and locking
- Arnoldi method with explicit restart and deflation

Also interfaces to external software such as ARPACK

Details: Specification of Options

`EPSSetFromOptions(EPS eps)`

Looks in the command line for options related to EPS

For example, the following command line

```
% program -eps_hermitian
```

is equivalent to a call `EPSSetProblemType(eps,EPS_HEP)`

Other options have an associated function call

```
% program -eps_nev 6 -eps_tol 1e-8
```

`EPSView(EPS eps, PetscViewer viewer)`

Prints information about the object (equivalent to `-eps_view`)

Details: Specification of Options

`EPSSetFromOptions(EPS eps)`

Looks in the command line for options related to EPS

For example, the following command line

```
% program -eps_hermitian
```

is equivalent to a call `EPSSetProblemType(eps,EPS_HEP)`

Other options have an associated function call

```
% program -eps_nev 6 -eps_tol 1e-8
```

`EPSView(EPS eps, PetscViewer viewer)`

Prints information about the object (equivalent to `-eps_view`)

Details: Specification of Options

`EPSSetFromOptions(EPS eps)`

Looks in the command line for options related to EPS

For example, the following command line

```
% program -eps_hermitian
```

is equivalent to a call `EPSSetProblemType(eps,EPS_HEP)`

Other options have an associated function call

```
% program -eps_nev 6 -eps_tol 1e-8
```

`EPSView(EPS eps, PetscViewer viewer)`

Prints information about the object (equivalent to `-eps_view`)

Run-Time Examples

```
% program -eps_view -eps_monitor

% program -eps_type power -eps_nev 6 -eps_ncv 24

% program -eps_type arnoldi -eps_tol 1e-8 -eps_max_it 2000

% program -eps_type subspace -eps_hermitian -log_summary

% program -eps_type lapack

% program -eps_type arpack -eps_plot_eigs -draw_pause -1

% program -eps_type blzpack -eps_smallest_real
```

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

- The user need not manage the ST object directly
- Internally, the eigensolver works with the operator T
- At the end, eigenvalues are transformed back automatically

ST	Standard problem	Generalized problem
`shift`	$A + \sigma I$	$B^{-1}A + \sigma I$
`sinvert`	$(A - \sigma I)^{-1}$	$(A - \sigma B)^{-1}B$
`cayley`	$(A - \sigma I)^{-1}(A + \tau I)$	$(A - \sigma B)^{-1}(A + \tau B)$

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

- The user need not manage the ST object directly
- Internally, the eigensolver works with the operator T
- At the end, eigenvalues are transformed back automatically

ST	Standard problem	Generalized problem
shift	$A + \sigma I$	$B^{-1}A + \sigma I$
sinvert	$(A - \sigma I)^{-1}$	$(A - \sigma B)^{-1}B$
cayley	$(A - \sigma I)^{-1}(A + \tau I)$	$(A - \sigma B)^{-1}(A + \tau B)$

Spectral Transformation in SLEPc

An ST object is always associated to any EPS object

$$Ax = \lambda x \quad \Longrightarrow \quad Tx = \theta x$$

- The user need not manage the ST object directly
- Internally, the eigensolver works with the operator T
- At the end, eigenvalues are transformed back automatically

ST	Standard problem	Generalized problem
`shift`	$A + \sigma I$	$B^{-1}A + \sigma I$
`sinvert`	$(A - \sigma I)^{-1}$	$(A - \sigma B)^{-1}B$
`cayley`	$(A - \sigma I)^{-1}(A + \tau I)$	$(A - \sigma B)^{-1}(A + \tau B)$

Accessing the ST Object

The user does not create the ST object

`EPSGetST(EPS eps, ST *st)`

Gets the ST object associated to an EPS

Necessary for setting options in the source code

Linear Solves. All operators contain an inverse (except $B^{-1}A + \sigma I$ in the case of a standard problem)

- Linear solves are handled internally via a KSP object

`STGetKSP(ST st, KSP *ksp)`

Gets the KSP object associated to an ST

All KSP options are available, by prepending the `-st_` prefix

Accessing the ST Object

The user does not create the ST object

`EPSGetST(EPS eps, ST *st)`

Gets the ST object associated to an EPS

Necessary for setting options in the source code

Linear Solves. All operators contain an inverse (except $B^{-1}A + \sigma I$ in the case of a standard problem)

- Linear solves are handled internally via a KSP object

`STGetKSP(ST st, KSP *ksp)`

Gets the KSP object associated to an ST

All KSP options are available, by prepending the `-st_` prefix

More Run-Time Examples

```
% program -eps_type power -st_type shift -st_shift 1.5

% program -eps_type power -st_type sinvert -st_shift 1.5

% program -eps_type power -st_type sinvert
          -eps_power_shift_type rayleigh

% program -eps_type arpack -eps_tol 1e-6
          -st_type sinvert -st_shift 1
          -st_ksp_type cgs -st_ksp_rtol 1e-8
          -st_pc_type sor  -st_pc_sor_omega 1.3
```

A Survey of Eigenproblems

Selecting the Portion of the Spectrum

Extreme eigenvalues:

- Dominant eigenvalues (e.g. principal component analyses)
- Rightmost eigenvalues (e.g. stability problems)
- Smallest eigenvalues (e.g. vibration analyses)

Interior eigenvalues:

- Eigenvalues closest to the scalar σ
- Eigenvalues closest to the imaginary axis

Other:

- All eigenvalues in interval $[a, b]$

Selecting the Portion of the Spectrum

Extreme eigenvalues:

- Dominant eigenvalues (e.g. principal component analyses)
- Rightmost eigenvalues (e.g. stability problems)
- Smallest eigenvalues (e.g. vibration analyses)

Interior eigenvalues:

- Eigenvalues closest to the scalar σ
- Eigenvalues closest to the imaginary axis

Other:

- All eigenvalues in interval $[a, b]$

Selecting the Portion of the Spectrum

Extreme eigenvalues:

- Dominant eigenvalues (e.g. principal component analyses)
- Rightmost eigenvalues (e.g. stability problems)
- Smallest eigenvalues (e.g. vibration analyses)

Interior eigenvalues:

- Eigenvalues closest to the scalar σ
- Eigenvalues closest to the imaginary axis

Other:

- All eigenvalues in interval $[a, b]$

Current Approach in SLEPc

`EPSSetWhichEigenpairs(EPS eps, EPSWhich which)`

Specifies which part of the spectrum is requested

`which`	Command line key	Sorting criterion
`EPS_LARGEST_MAGNITUDE`	`-eps_largest_magnitude`	Largest $\lvert\lambda\rvert$
`EPS_SMALLEST_MAGNITUDE`	`-eps_smallest_magnitude`	Smallest $\lvert\lambda\rvert$
`EPS_LARGEST_REAL`	`-eps_largest_real`	Largest $\mathrm{Re}(\lambda)$
`EPS_SMALLEST_REAL`	`-eps_smallest_real`	Smallest $\mathrm{Re}(\lambda)$
`EPS_LARGEST_IMAGINARY`	`-eps_largest_imaginary`	Largest $\mathrm{Im}(\lambda)$
`EPS_SMALLEST_IMAGINARY`	`-eps_smallest_imaginary`	Smallest $\mathrm{Im}(\lambda)$

- Eigenvalues are sought according to this criterion (not all possibilities available for all solvers)
- Interior eigenvalues computation supported via spectral transform

Computational Interval

It is convenient in some applications to specify a computational interval $[a, b]$

Accept only solutions inside (or outside) the interval $[a, b]$

- Easy to implement
- Internally keep track of converged unwanted eigenpairs

Compute *all* eigenvalues in the interval $[a, b]$

- Requires specialized eigensolver, e.g. Lanczos with spectrum slicing
- Requires computation of inertia for different shifts

Computational Interval

It is convenient in some applications to specify a computational interval $[a, b]$

Accept only solutions inside (or outside) the interval $[a, b]$

- Easy to implement
- Internally keep track of converged unwanted eigenpairs

Compute *all* eigenvalues in the interval $[a, b]$

- Requires specialized eigensolver, e.g. Lanczos with spectrum slicing
- Requires computation of inertia for different shifts

Computational Interval

It is convenient in some applications to specify a computational interval $[a, b]$

Accept only solutions inside (or outside) the interval $[a, b]$

- Easy to implement
- Internally keep track of converged unwanted eigenpairs

Compute *all* eigenvalues in the interval $[a, b]$

- Requires specialized eigensolver, e.g. Lanczos with spectrum slicing
- Requires computation of inertia for different shifts

Preserving the Symmetry

In the case of generalized eigenproblems in which both A and B are symmetric, symmetry is lost because none of $B^{-1}A + \sigma I$, $(A - \sigma B)^{-1}B$ or $(A - \sigma B)^{-1}(A + \tau B)$ is symmetric

Choice of Inner Product

- Standard Hermitian inner product: $\langle x, y \rangle = x^H y$
- B-inner product: $\langle x, y \rangle_B = x^H B\, y$

Observations:

- $\langle x, y \rangle_B$ is a genuine inner product only if B is symmetric positive definite
- $\mathbb{R}^n$ with $\langle x, y \rangle_B$ is isomorphic to the Euclidean n-space $\mathbb{R}^n$ with the standard Hermitian inner product
- $B^{-1}A$ is auto-adjoint with respect to $\langle x, y \rangle_B$

Preserving the Symmetry

In the case of generalized eigenproblems in which both A and B are symmetric, symmetry is lost because none of $B^{-1}A + \sigma I$, $(A - \sigma B)^{-1}B$ or $(A - \sigma B)^{-1}(A + \tau B)$ is symmetric

Choice of Inner Product

- Standard Hermitian inner product: $\langle x, y \rangle = x^H y$
- B-inner product: $\langle x, y \rangle_B = x^H B\, y$

Observations:

- $\langle x, y \rangle_B$ is a genuine inner product only if B is symmetric positive definite
- $\mathbb{R}^n$ with $\langle x, y \rangle_B$ is isomorphic to the Euclidean n-space $\mathbb{R}^n$ with the standard Hermitian inner product
- $B^{-1}A$ is auto-adjoint with respect to $\langle x, y \rangle_B$

Preserving the Symmetry

In the case of generalized eigenproblems in which both A and B are symmetric, symmetry is lost because none of $B^{-1}A + \sigma I$, $(A - \sigma B)^{-1}B$ or $(A - \sigma B)^{-1}(A + \tau B)$ is symmetric

Choice of Inner Product

- Standard Hermitian inner product: $\langle x, y \rangle = x^H y$
- B-inner product: $\langle x, y \rangle_B = x^H B\, y$

Observations:

- $\langle x, y \rangle_B$ is a genuine inner product only if B is symmetric positive definite
- $\mathbb{R}^n$ with $\langle x, y \rangle_B$ is isomorphic to the Euclidean n-space $\mathbb{R}^n$ with the standard Hermitian inner product
- $B^{-1}A$ is auto-adjoint with respect to $\langle x, y \rangle_B$

Complex Symmetric Problems

Consider the complex symmetric eigenvalue problem

$$Ax = \lambda x \;, \quad A = A^T \in \mathbb{C}^{n\times n}$$

Lanczos Method for Complex Symmetric Problems

- Build a complex orthogonal set of vectors: $V_j^T V_j = I_j$
- Obtain a complex symmetric tridiagonal matrix $T_j = T_j^T$

Lanczos can be applied if replacing the standard Hermitian inner product by the indefinite bilinear form $\langle x, y\rangle = x^T y$

- Breakdown occurs if $\hat{v}_{j+1}^T \hat{v}_{j+1} = 0$ but $\hat{v}_{j+1} \neq 0$

Complex Symmetric Problems

Consider the complex symmetric eigenvalue problem

$$Ax = \lambda x \ , \quad A = A^T \in \mathbb{C}^{n \times n}$$

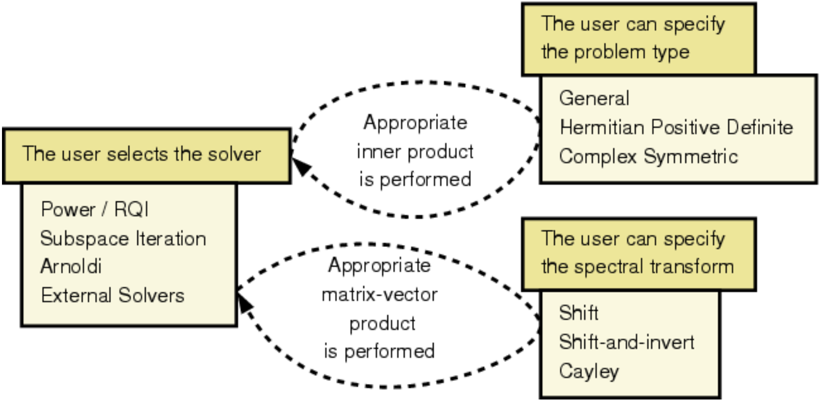
Lanczos Method for Complex Symmetric Problems

- Build a complex orthogonal set of vectors: $V_j^T V_j = I_j$
- Obtain a complex symmetric tridiagonal matrix $T_j = T_j^T$

Lanczos can be applied if replacing the standard Hermitian inner product by the indefinite bilinear form $\langle x, y \rangle = x^T y$

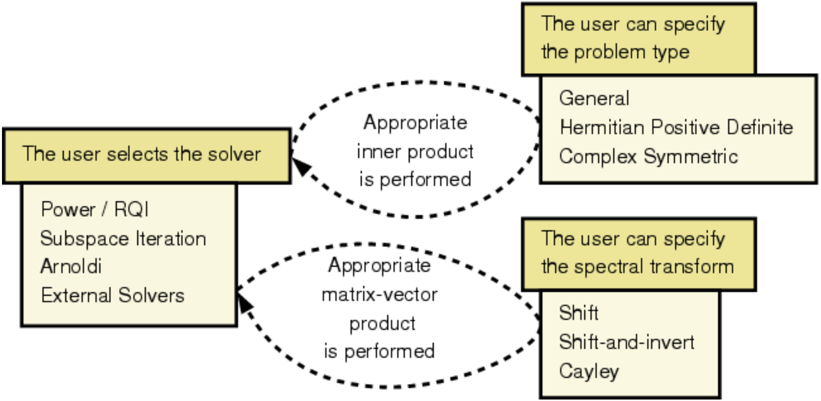
- Breakdown occurs if $\hat{v}_{j+1}^T \hat{v}_{j+1} = 0$ but $\hat{v}_{j+1} \neq 0$

SLEPc Abstraction



These operations are virtual functions: STInnerProduct and STApply

SLEPc Abstraction



These operations are virtual functions: STInnerProduct and STApply

Example: Computational Electromagnetics

Objective: Analysis of resonant cavities

Source-free wave equations

$$\nabla \times (\hat{\mu}_r^{-1} \nabla \times \vec{E}) - \kappa_0^2 \hat{\varepsilon}_r \vec{E} = 0$$
$$\nabla \times (\hat{\varepsilon}_r^{-1} \nabla \times \vec{H}) - \kappa_0^2 \hat{\mu}_r \vec{H} = 0$$

Target: A few smallest nonzero eigenfrequencies

Discretization: 1^{st} order edge finite elements (tetrahedral)

$$Ax = \kappa_0^2 Bx$$

Generalized Eigenvalue Problem

- A and B are large and sparse, possibly complex
- A is (complex) symmetric and semi-positive definite
- B is (complex) symmetric and positive definite

Example: Computational Electromagnetics

Objective: Analysis of resonant cavities

Source-free wave equations

$$\nabla \times (\hat{\mu}_r^{-1} \nabla \times \vec{E}) - \kappa_0^2 \hat{\varepsilon}_r \vec{E} = 0$$
$$\nabla \times (\hat{\varepsilon}_r^{-1} \nabla \times \vec{H}) - \kappa_0^2 \hat{\mu}_r \vec{H} = 0$$

Target: A few smallest nonzero eigenfrequencies

Discretization: 1^{st} order edge finite elements (tetrahedral)

$$Ax = \kappa_0^2 Bx$$

Generalized Eigenvalue Problem

- A and B are large and sparse, possibly complex
- A is (complex) symmetric and semi-positive definite
- B is (complex) symmetric and positive definite

Example: Computational Electromagnetics (cont'd)

Matrix A has a high-dimensional null space, $\mathcal{N}(A)$

- The problem $Ax = \kappa_0^2 Bx$ has many zero eigenvalues
- These eigenvalues should be avoided during computation

$$\underbrace{\lambda_1, \lambda_2, \ldots, \lambda_k}_{=0}, \underbrace{\lambda_{k+1}, \lambda_{k+2}}_{\text{Target}}, \ldots, \lambda_n$$

Eigenfunctions associated to 0 are irrotational electric fields, $\vec{E} = -\nabla\Phi$. This allows the computation of a basis of $\mathcal{N}(A)$

Constrained Eigenvalue Problem

$$\left.\begin{array}{c} Ax = \kappa_0^2 Bx \\ C^T Bx = 0 \end{array}\right\}$$

where the columns of C span $\mathcal{N}(A)$

Example: Computational Electromagnetics (cont'd)

Matrix A has a high-dimensional null space, $\mathcal{N}(A)$

- The problem $Ax = \kappa_0^2 Bx$ has many zero eigenvalues
- These eigenvalues should be avoided during computation

$$\underbrace{\lambda_1, \lambda_2, \ldots, \lambda_k}_{=0}, \underbrace{\lambda_{k+1}, \lambda_{k+2}}_{\text{Target}}, \ldots, \lambda_n$$

Eigenfunctions associated to 0 are irrotational electric fields, $\vec{E} = -\nabla\Phi$. This allows the computation of a basis of $\mathcal{N}(A)$

Constrained Eigenvalue Problem

$$\left.\begin{aligned} Ax &= \kappa_0^2 Bx \\ C^T Bx &= 0 \end{aligned}\right\}$$

where the columns of C span $\mathcal{N}(A)$

Deflation Subspaces

EPSAttachDeflationSpace(EPS eps,int n,Vec *ds,PetscTruth ortho)

Allows to provide a basis of a deflating subspace $\mathcal{S}$

The eigensolver works with the restriction of the problem to the orthogonal complement of this subspace $\mathcal{S}$

Possible uses:

- When $\mathcal{S}$ is an invariant subspace, then the corresponding eigenpairs are not computed again
- If $\mathcal{S}$ is the null space of the operator, then zero eigenvalues are skipped
- In general, for constrained eigenvalue problems
- Also for singular pencils (A and B share a common null space)

Deflation Subspaces

EPSAttachDeflationSpace(EPS eps,int n,Vec *ds,PetscTruth ortho)

Allows to provide a basis of a deflating subspace $\mathcal{S}$

The eigensolver works with the restriction of the problem to the orthogonal complement of this subspace $\mathcal{S}$

Possible uses:

- When $\mathcal{S}$ is an invariant subspace, then the corresponding eigenpairs are not computed again
- If $\mathcal{S}$ is the null space of the operator, then zero eigenvalues are skipped
- In general, for constrained eigenvalue problems
- Also for singular pencils (A and B share a common null space)

Quadratic Eigenvalue Problem (QEP)

In applications such as the analysis of damped vibrating systems:

$$(A\lambda^2 + B\lambda + C)x = 0$$

Transform the problem to a generalized eigenproblem by increasing the order of the system, e.g. defining $v = [\lambda x, x]^T$

$$\begin{bmatrix} -B & -C \\ I & 0 \end{bmatrix} v = \lambda \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} v$$

PETSc's *shell* matrices can be used to represent blocked matrices

Quadratic Eigenvalue Problem (QEP)

In applications such as the analysis of damped vibrating systems:

$$(A\lambda^2 + B\lambda + C)x = 0$$

Transform the problem to a generalized eigenproblem by increasing the order of the system, e.g. defining $v = [\lambda x, x]^T$

$$\begin{bmatrix} -B & -C \\ I & 0 \end{bmatrix} v = \lambda \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} v$$

PETSc's *shell* matrices can be used to represent blocked matrices

Singular Value Decomposition (SVD)

Given $A \in \mathbb{R}^{m\times n}$, compute orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ such that

$$U^T A\,V = \mathrm{diag}(\sigma_1,\ldots,\sigma_p)$$

with $p = \min\{m,n\}$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$

Equivalent eigenvalue problems:

$$A^T A\,v_i = \sigma_i^2 v_i$$

Poor accuracy for small σ_i's

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \sigma_i \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Again, *shell* matrices can be used (template example `ex8.c`)

Singular Value Decomposition (SVD)

Given $A \in \mathbb{R}^{m\times n}$, compute orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ such that

$$U^T A\, V = \mathrm{diag}(\sigma_1, \ldots, \sigma_p)$$

with $p = \min\{m, n\}$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$

Equivalent eigenvalue problems:

$$A^T A\, v_i = \sigma_i^2 v_i$$

Poor accuracy for small σ_i's

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \sigma_i \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Again, *shell* matrices can be used (template example `ex8.c`)

Singular Value Decomposition (SVD)

Alternative to the previous approaches:

Lanczos Bidiagonalization (Golub and Kahan, 1965)

- Two orthogonal sets of Lanczos vectors: u_j and v_j
- Three-term recurrences associated to A and A^T
- Lower bidiagonal matrix B_j

Possible implementation in SLEPc

- Specialized solver associated to the `lanczos` eigensolver
- Used when `ProblemType` = `EPS_SVD`
- Use a templated scheme if no specialized solver is available

Singular Value Decomposition (SVD)

Alternative to the previous approaches:

Lanczos Bidiagonalization (Golub and Kahan, 1965)

- Two orthogonal sets of Lanczos vectors: u_j and v_j
- Three-term recurrences associated to A and A^T
- Lower bidiagonal matrix B_j

Possible implementation in SLEPc

- Specialized solver associated to the `lanczos` eigensolver
- Used when `ProblemType` = `EPS_SVD`
- Use a templated scheme if no specialized solver is available

Example: Nuclear Engineering

Modal analysis of nuclear reactor cores

Objectives:

- Improve safety
- Reduce operation costs

Lambda Modes Equation

$$\mathcal{L}\phi = \frac{1}{\lambda}\mathcal{M}\phi$$

Target: modes associated to largest λ

- Criticality (eigenvalues)
- Prediction of instabilities and transient analysis (eigenvectors)

SLEPc

Example: Nuclear Engineering

Modal analysis of nuclear reactor cores

Objectives:

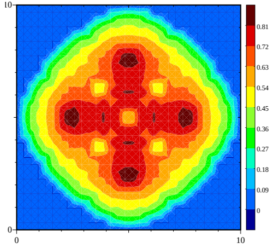
- Improve safety
- Reduce operation costs

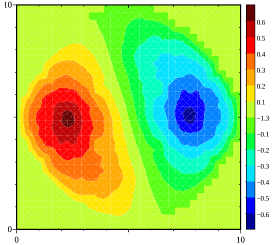
Lambda Modes Equation

$$\mathcal{L}\phi = \frac{1}{\lambda}\mathcal{M}\phi$$

Target: modes associated to largest λ

- Criticality (eigenvalues)
- Prediction of instabilities and transient analysis (eigenvectors)





Example: Nuclear Engineering (cont'd)

Discretized eigenproblem

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

Can be restated as

$$N\psi_1 = \lambda L_{11}\psi_1 \ , \quad N = M_{11} + M_{12}L_{22}^{-1}L_{21}$$

- Generalized eigenvalue problem
- Matrix N should not be computed explicitly
- The *adjoint problem* has to be solved also: this amounts to computing the left eigenvectors

Example: Nuclear Engineering (cont'd)

Discretized eigenproblem

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

Can be restated as

$$N\psi_1 = \lambda L_{11}\psi_1 \;, \quad N = M_{11} + M_{12}L_{22}^{-1}L_{21}$$

- Generalized eigenvalue problem
- Matrix N should not be computed explicitly
- The *adjoint problem* has to be solved also: this amounts to computing the left eigenvectors

Example: Nuclear Engineering (cont'd)

Discretized eigenproblem

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}$$

Can be restated as

$$N\psi_1 = \lambda L_{11}\psi_1 \ , \quad N = M_{11} + M_{12}L_{22}^{-1}L_{21}$$

- Generalized eigenvalue problem
- Matrix N should not be computed explicitly
- The *adjoint problem* has to be solved also: this amounts to computing the left eigenvectors

Left Invariant Subspaces

Can be computed with Two-sided Eigensolvers

- Non-symmetric Lanczos
- Two sets of Lanczos vectors: v_j (right) and w_j (left)
- Bi-orthogonality condition: $W_j^T V_j = I_j$
- Also two-sided variants of other eigensolvers: RQI, Arnoldi, JD

Proposed implementation in SLEPc

- EPS attribute `SolverClass = { EPS_ONE_SIDE, EPS_TWO_SIDE}`
- Requirements: Storage for left vectors and $y = A^T x$ operation

Other applications

- Reduced-order modeling of Linear Time-Invariant (LTI) systems with multiple inputs and outputs

SLEPc

Left Invariant Subspaces

Can be computed with Two-sided Eigensolvers

- Non-symmetric Lanczos
- Two sets of Lanczos vectors: v_j (right) and w_j (left)
- Bi-orthogonality condition: $W_j^T V_j = I_j$
- Also two-sided variants of other eigensolvers: RQI, Arnoldi, JD

Proposed implementation in SLEPc

- EPS attribute `SolverClass = { EPS_ONE_SIDE, EPS_TWO_SIDE}`
- Requirements: Storage for left vectors and $y = A^T x$ operation

Other applications

- Reduced-order modeling of Linear Time-Invariant (LTI) systems with multiple inputs and outputs

Left Invariant Subspaces

Can be computed with Two-sided Eigensolvers

- Non-symmetric Lanczos
- Two sets of Lanczos vectors: v_j (right) and w_j (left)
- Bi-orthogonality condition: $W_j^T V_j = I_j$
- Also two-sided variants of other eigensolvers: RQI, Arnoldi, JD

Proposed implementation in SLEPc

- EPS attribute `SolverClass = { EPS_ONE_SIDE, EPS_TWO_SIDE}`
- Requirements: Storage for left vectors and $y = A^T x$ operation

Other applications

- Reduced-order modeling of Linear Time-Invariant (LTI) systems with multiple inputs and outputs

More advanced eigensolvers

Block/band algorithms

- Support for multi-vectors in PETSc is quite basic
- Efficiency gain may be moderate

Restart techniques

- Implicit restart (Sorensen, 1992)
- Krylov-Schur restart (Stewart, 2001)

Preconditioned eigensolvers

- Jacobi-Davidson, Preconditioned Conjugate Gradient, Preconditioned Lanczos, ...
- Need to figure out how these can coexist with ST

Multilevel eigensolvers: AMLS

Higher Algorithmic Level

Some applications require to solve many successive eigenproblems

- Family of slightly perturbed eigenproblems
- Nonlinear or parameter dependent eigenvalue problems
- Eigenpath continuation (e.g. bifurcation analysis)

Potentially useful high-level algorithmic schemes:

- Initial approximation to the solution
  - Single initial vector is not sufficient
  - Krylov recycling: reuse all the information available from previous problem
- Homotopy method for eigenpath continuation
  - Especially useful in the case of symmetric problems

Higher Algorithmic Level

Some applications require to solve many successive eigenproblems

- Family of slightly perturbed eigenproblems
- Nonlinear or parameter dependent eigenvalue problems
- Eigenpath continuation (e.g. bifurcation analysis)

Potentially useful high-level algorithmic schemes:

- Initial approximation to the solution
  - Single initial vector is not sufficient
  - Krylov recycling: reuse all the information available from previous problem
- Homotopy method for eigenpath continuation
  - Especially useful in the case of symmetric problems

Concluding Remarks

Concluding Remarks

SLEPc 2.2.1 (current version)

- General library for the solution of eigenvalue problems
- Basic eigensolvers plus spectral transformation
- Flexibility that enables to solve (not-so-simple) standard and generalized problems

SLEPc 2.2.2 (next release)

- Support for two-sided eigensolvers
- Block/band variants of some solvers
- Partial support for computational intervals

Future directions

- More modern eigensolvers (implicit restart, preconditioned, ...)

Concluding Remarks

SLEPc 2.2.1 (current version)

- General library for the solution of eigenvalue problems
- Basic eigensolvers plus spectral transformation
- Flexibility that enables to solve (not-so-simple) standard and generalized problems

SLEPc 2.2.2 (next release)

- Support for two-sided eigensolvers
- Block/band variants of some solvers
- Partial support for computational intervals

Future directions

- More modern eigensolvers (implicit restart, preconditioned, ...)

Concluding Remarks

SLEPc 2.2.1 (current version)

- General library for the solution of eigenvalue problems
- Basic eigensolvers plus spectral transformation
- Flexibility that enables to solve (not-so-simple) standard and generalized problems

SLEPc 2.2.2 (next release)

- Support for two-sided eigensolvers
- Block/band variants of some solvers
- Partial support for computational intervals

Future directions

- More modern eigensolvers (implicit restart, preconditioned, ...)

Thanks!

http://www.grycap.upv.es/slepc

slepc-maint@grycap.upv.es